```
MODULE Main ;

IMPORT IO ;
FROM Aufgabe35 IMPORT Random ;

(* Variante von BucketSort, die in situ arbeitet, dafuer aber nicht
   stabil ist.  Ausser dem Histogramm wird nur Speicher O(1)
   benoetigt.  Idee:
   1. Erzeuge Histogramm
   2. Summiere Histogramm zu Bucket-Obergrenzen auf.
   3. Sortiere Eingabefeld in situ durch Ringtausch:
          Fuer jeden Bucket k = 1, 2,... :
              Solange noch Elemente > k im Bucket sind:
                  Fuehre Ringtausch durch bis Ausgangsposition wieder erreicht
      M.a.W.: Der Reihe nach wird jeder Bucket von fremden Elementen
      befreit.  Da Elemente < k nicht mehr vorhanden sein koennen,
      kann dies als Abbruchbedinung verwendet werden, so dass man
      keine Bucket- Untergrenzen speichern muss. *)


TYPE Key           = [1..5] ;  (* zulaessige Schluesselmenge *)
     KeyOrSentinel = [0..5] ;  (* Schluesselmenge mit Sentinelelement *)


PROCEDURE BucketSort(VAR a: ARRAY OF KeyOrSentinel) =
VAR count := ARRAY Key OF INTEGER {0,..} ;
    u, v : Key ;
BEGIN
    a[0] := FIRST(KeyOrSentinel) ; (* Sentinel *)

    (* Histogramm erzeugen *)
    FOR i := 1 TO LAST(a) DO
        INC(count[a[i]]) ;
    END ;

    (* Histogramm in Indizes umrechnen *)
    FOR k := FIRST(Key) + 1 TO LAST(Key) DO
        count[k] := count[k] + count[k-1] ;
    END ;

    (* Sortieren *)
    FOR k := FIRST(Key) TO LAST(Key) DO
        WHILE a[count[k]] >= k DO
            v := a[count[k]] ;
            REPEAT (* Ringtausch *)
                u := v ;
                v := a[count[u]] ;
                a[count[u]] := u ;
                DEC(count[u]) ;
            UNTIL u = k ;
        END ;
    END ;
END BucketSort ;


VAR data : ARRAY [0..50] OF KeyOrSentinel ;
BEGIN
    FOR i := 1 TO LAST(data) DO
        data[i] := VAL(Random(5), Key) ;
        IO.PutInt(data[i]) ; IO.Put("\n") ;
    END ;
    IO.Put("\n") ;

    BucketSort(data) ;

    FOR i := 1 TO LAST(data) DO
        IO.PutInt(data[i]) ; IO.Put("\n") ;
    END ;
END Main .
```